05 月 16 日

第四章：当 Ulysses 遇上 URL Schemes

| 本文为付费栏目文章，您已订阅，可阅读全文 |

想要高效地使用 iOS 应用，就不得不依靠 URL Schemes 的帮助。

不少人喜欢用 Drafts、Launch Center Pro 和 Workflow 等工具。**将这些效率工具与 Ulysses 相结合**，正是 URL Schemes 最能帮助 Ulysses for iOS 提高效率的地方。

注：以下内容需要对 URL Schemes 有一定的了解。如果你之前完全没接触过，希望先阅读 Hum 写的《URL Schemes 使用详解》一文，再来回过来阅读本章的内容。

初探 Ulysses 中的 URL Schemes

有的应用所提供的 URL Schemes，根据功能不同语句的格式也有所不同。而在 Ulysses 中使用的 URL Schemes，统一按照下列的格式书写：

```
ulysses://x-callback-url/[action]?[x-callback parameters]&[parameters]
```

`ulysses://` 不必多说，自然是 URL Scheme 中的指明语句所属应用的部分。

看到句中 `x-callback-url`，这意味着 Ulysses 支持在 URL Schemes 中使用 x-callback-urls。支持 x-callback-urls 意味着我们可以实现一些更复杂的操作，例如在应用间跳转，将多个动作串联起来。

而 Ulysses 支持 x-callback-urls 的目的不仅如此。x-callback-urls 的另一个好处，就是提供一种**标准化的 URL Schemes 语法结构**。URL Schemes 的最后三部分，就是最重要的三个参数：

- `[action]`：指明这条 URL Scheme 想要执行的操作，例如打开文稿、新建文稿、添加注释等。这部分属于**必填**的内容；
- `[x-callback parameters]`：x-callback-url 定义了几种特殊的的参数，用来实现与 x-callback-url 相关的功能，例如 `x-success` 可以在动作

执行成功后跳转回发起 URL Schemes 的应用。这部分属于**选填**的内容；

- `[parameters]`：指明 action 所需要的内容。例如是打开文稿的 URL Scheme，自然要给出文稿的内容。这部分属于**必填**的内容。

从上面的标准 URL Scheme 语句可以观察出，我们不需要逐一了解 Ulysses 的每一条 URL Scheme 该怎么写，只需要学会 `[action]`、`[x-callback parameters]` 和 `[parameters]` 三个参数的用法。搞明白每个参数能填写哪些内容，自然能举一反三，掌握 Ulysses 所有的 URL Schemes 用法。

Action

最值得关注的是 `[action]`。它是 URL Scheme 中的核心要素，通过它可以看出这句 **URL Scheme 的主要目的**。

例如当我们想创建一份新的文稿时，就需要使用 `new-sheet`；想在文库中创建一个新组，就使用 `new-group`；想在现有的文稿中添加一段话，就使用 `insert`。在 Ulysses 的官方文档 中，一共列出了 Ulysses 所支持的 22 种 `[action]`。根据它们的使用场景，我将它们大致分为以下五类：

- 创建新内容：new–sheet、new–group
- 修改现有文稿：insert、attach–note、update–note、remove–note、attach–image、attach–keywords、remove–keywords、set–group–title、set–sheet–title
- 整理文库：move、copy、trash
- 导航到特定文稿或租：open、open–all、open–recent、open–favorites
- 获取文稿元数据：get–item、get–root–items、read–sheet、get–quick–look–url

通过它们的名字和在 Ulysses 的官方文档 中的介绍，就能理解好大部分 action 的含义。不过有些少见的 action，我们在这里特别解释一下。

`attach-note`、`update-note`、`remove-note` 中的「note」，指的不是书写文稿时的评论或是注解，而是附件窗口中的注释，故而只能对 iCloud 文库中的文稿使用。

`get-item` 和 `get-root-items` 这两个动作可能在其它应用的 URL Schemes 中很少见，`get-item` 会返回文稿或者组的元数据，例如标题、创建时间、修改时间或者类型等信息。`get-root-items` 则有所不同，它返回的是整个文库的信息。所有数据将会以 JSON 格式返回。

parameters

而**对应每一种 `[action]`，能用的 `[parameters]` 都会随之变化**。`[parameters]` 可以看作执行某个 `[action]` 时想表达的具体内容。

常见参数的用法

就以最常见的 `new-sheet` 为例。想创建一份新的文稿，我们当然要考虑文稿的具体内容是什么，在哪个组，是什么格式等问题。对此 `new-sheet` 提供了四种参数可以设置：`text`、`group`、`format` 和 `index`。

- `text` 就是将被写进文稿的内容。注意在 URL Scheme 语句中，`text` 对应的内容必须是用 URL 编码的。

- `group` 则是文稿所保存的组。如果是嵌套中的子组，就用 `/` 来分隔，例如 `/My Group/My Subgroup`。如果不设定特别的组，就默认被收进收件箱中。
- `format` 用来设定文稿的格式，可以从 markdown、text 和 html 中选择。如果不设定，就默认保存为 markdown 格式的文稿。
- `index` 指明了文稿在组中的排序位置，和其它编程语言一样，排序从零开始计数，也就是说 `index=0` 才表示排在第一位。

最后将这四种参数组合起来（不一定每种都要用上），用 `&` 连接；动作和参数之间用 `?` 号连接。一句 URL Scheme 就完成了。以下面这条 URL Scheme 为例，就使用了 `new-sheet` 动作，`text` 参数设定为「Demo」，并且通过 `index` 参数将它排在收件箱的第一位：

```
ulysses://x-callback-url/new-sheet?text=Demo&index=0
```

创建新文稿

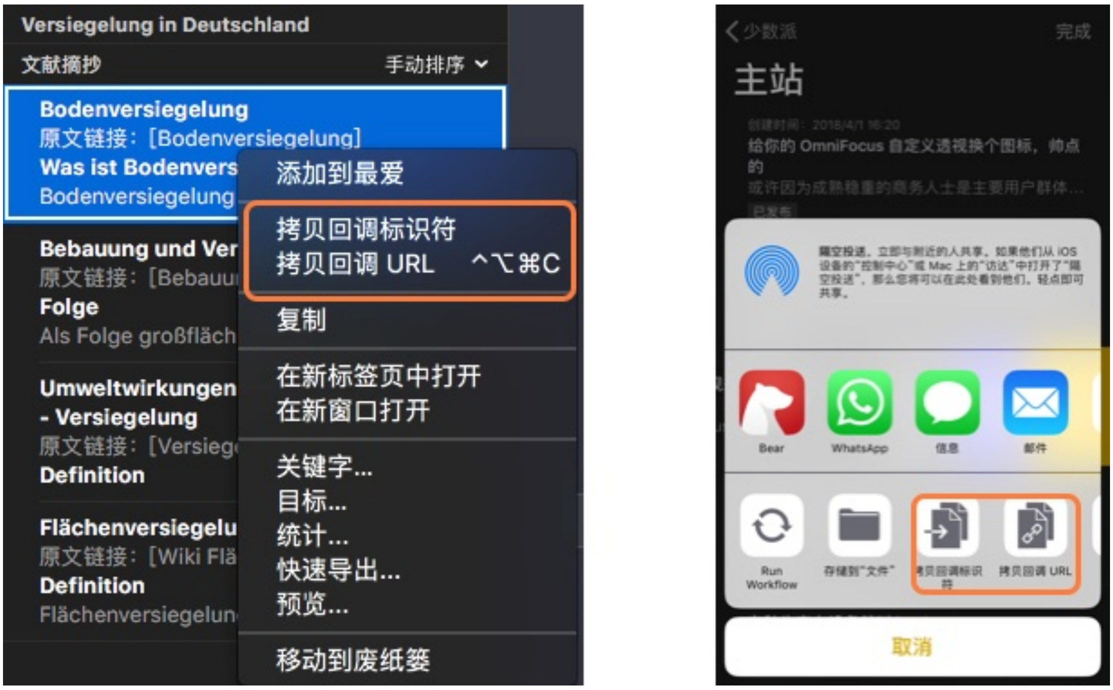
全部 22 种 `[action]` 对应的参数都在官方文档中有详细解释和举例，绝大多数参数都非常易懂易用。但还有一些特殊的参数需要特别说明下。

id

在很多涉及到具体文稿或者组的 `action` 中，例如 `insert`、`attach-note` 和 `move`，都必须输入 `id` 这个参数。

`id` 可以理解为每个**文稿的身份证号**。如果我们想编辑修改一个文稿，最直接准确就是先告诉 `action`，它所处理的文稿 `id` 是多少，这样才能在文库中找到对应的文稿，进行下一步操作。

想获得文稿（或者组）的 `id`，就需要进入 Ulysses。在 Ulysses for Mac 中，按住「⌥ 」再右键点击文稿，就能看到「拷贝回调标识符」选项。而在 Ulysses for iOS 中，向左滑动，点击「更多 – 分享」，就能看到相同的选项。


Versiegelung in Deutschland
文献摘抄
手动排序
Bodenversiegelung
原文链接：[Bodenversiegelung]
Was ist Bodenvers
Bodenversiegelung
Bebauung und Ver
原文链接：[Bebauu
Folge
Als Folge großfläch
Umweltwirkungen
- Versiegelung
原文链接：[Versieg
Definition
Flächenversiegelu
原文链接：[Wiki Flä
Definition
Flächenversiegelun
添加到最爱
拷贝回调标识符
拷贝回调 URL ^⌥⌘C
复制
在新标签页中打开
在新窗口打开
关键字...
目标...
统计...
快速导出...
预览...
移动到废纸篓
少数派
完成
主站
Bear
WhatsApp
信息
邮件
Run Workflow
存储到"文件"
拷贝回调标识符
拷贝回调 URL
取消

获取回调标识符

小贴士：如果选择拷贝回调 URL，就能直接获得打开这篇文稿的 URL Scheme。

有了文稿的 `id`，就能使用 `action` 来编辑文稿。虽然通常我不会在 Ulysses 之外，通过 URL Schemes 的方式大段的修改文章内容，但是时常在通勤路上，能想到该如何修改文章，或者添加哪些内容，这个时候就需要给它添加注释，方便坐下来后能打开 Ulysses 好好修改。

例如上文添加的演示文稿的 `id` 是 `_UmGl2zv_xdLdPaoe0VHkQ`，就能通过下列语句来给它添加一条新的注释：

```
ulysses://x-callback-url/attach-note?id=_UmGl2zv_xdLdPaoe0VHkQ&text=My new note
```

在 Launch Center Pro 中，需要用双层花括号「{{}}」将内容包裹起来，写成 `ulysses://x-callback-url/attach-note?id=_UmGl2zv_xdLdPaoe0VHkQ&text={{My new note}}`。

不过更方便的是，使用参数 `[prompt]`，让 Launch Center Pro 弹出输入框方便输入更细节的描述。

```
ulysses://x-callback-url/attach-note?id=_UmGl2zv_xdLdPaoe0VHkQ&text=[prompt]
```

给文稿添加新注释

虽然获取 id 的方式并不复杂，但是这意味着至少要进入一次 Ulysses。如果不想这么麻烦，可以**用路径来代替 id 来定位**。不过这种方法只能用于组，文稿是不能通过路径来定位的。

使用方法也很简单，直接指明文件夹的位置即可。不同层级间的文件夹名用斜线 `/`（注意是英文符号）隔开，但是如果只打开顶层文件夹则可以省略。例如我们要打开「少数派」，就可以写成：

```
ulysses://x-callback-url/open?id=少数派
```

而如果是「少数派」中的「主站」文件夹，就得写成：

```
ulysses://x-callback-url/open?id=/少数派/主站
```

打开 Ulysses 中的组

access-token

另一个需要留意的参数是 `access-token`，准确的说这不是一个可以由我们指定的参数，而是和 `id` 类似，需要针对特定软件获取的内容。

Ulysses 开放了如此多 `[action]`，试想一下，如果有其它应用想恶意篡改你文库中的文稿，通过 URL Schemes 就可以实现。为了保护 Ulysses 中的文稿，在进行一些「高危」的操作（例如 `remove-note`），就需要**对外部应用进行授权**，由你来控制哪些应用可以接触到 Ulysses 的文稿。

授权的原理在官方文档中有详细介绍，授权过程最终目的，是生成每个应用独有的 access-token。这个 access-token 就像是 Ulysses 给其它应用的一把钥匙，有了它，这个应用就可以打开 Ulysses 的文库。

我创建了一个简单 Workflow，方便我们快速获取应用的 access-token。以 Launch Center Pro 为例，运行 Workflow 后输入应用名「Launch」，access-token 就会显示出来并拷贝到剪贴板中。

获取 access-token

之前已经介绍过如何添加一个注释，可能添加完之后觉得当时的想法有欠考虑，我们就来尝试创建一个删除文稿中注释的 URL Scheme：

```
ulysses://x-callback-url/remove-note?id=_UmGl2zv_xdLdPaoe0VHkQ&index=0&
access-token=660df64d7da14fd78c7950d32360c1d4
```

在 `access-token` 后填上 Workflow 生成的 access-token，动作就能正确运行。因为每个应用需要自己的 access-token，这条 URL Scheme 放到其它应用中就不能正确运行。

删除注释

至于 x-callback parameters 则是标准的 `x-success` 和 `x-error`，分别用于动作运行成功和失败后的后续动作。在 Hum 写的《URL Schemes 使用详解》的对它们的用法有详细介绍，这里不再赘述。

在写作中使用 URL Schemes

完整的了解过 Ulysses 中 URL Schemes 的原理后，我们来尝试将它们用于实际场景中。

不过在这之前，我们先来思考下，如果将写作大致分成**收集素材、草稿、编辑和发布**这四步，URL Schemes 该用在写作的哪些环节中。

id 和 access-token 是限制运用 URL Schemes 的主要原因。

Ulysses 作为写作应用，文库的数据形式决定了文稿的 id 的重要性。当我们想修改文稿时，必须要通过 id 指明文稿，而想获得 id 又必须打开

Ulysses。由此一来，利用 URL Schemes 修改文稿的便利性大大降低。如果你已经打开了 Ulysses，为什么又要特定转到其它应用中使用 URL Schemes 来修改文稿呢?

当然这绝不意味着 Ulysses 中 URL Schemes 一无是处。在**写作的收集素材和草稿环节**，结合 Workflow 、Drafts 等工具，才是 URL Schemes 发挥作用的地方。

收集素材时使用 URL Schemes

在前文中介绍过利用 Drag & Drop 来抓取网页上的内容，不过那是针对仔细阅读、从中摘取精华的场景。如果我们想像印象笔记一样，**先将全文收集起来**，有待日后再整理，这种方法就不太高效。

使用 Web Page to Ulysses 这个 Workflow，就可以直接将网页全部收集到 Ulysses 中。

从 Workflow 1.5 开始，就内置了许多针对 Ulysses 的动作。不过观察后发现，基本上都是和 Ulysses 提供的动作相同。不过 Workflow 的优势就在于

能串联多个应用和动作，我们只用将这些 Ulysses 相关的动作当作 Workflow 的起点或终点，发挥 Workflow 「自动化执行动作」的特性。

注：如果你对 Workflow 完全不了解，可以先阅读 Hum 写的《Workflow 教程》。

Web Page to Ulysses 的实现原理其实非常直接，利用 `Get Contents of URL` 获取网页内容后，根据用户的选择转换格式，最后利用 `new-sheet` （Workflow 显示为 New Ulysses Sheet）动作将网页正文导入到 Ulysses 中。

利用 Web Page to Ulysses 收集素材

书写初稿时使用 URL Schemes

虽然我认为 Ulysses 本身就能完成从初稿到定稿的完整流程，但还有不少用户，喜欢将 Drafts 作为他在 iOS 设备上的唯一输入环境，再分发到其它应用中。

关于 Drafts 本身的技巧，本文不多赘述，感兴趣的读者可以去少数派上阅读。这里以 Sheet in Ulysses 动作为例。

Sheet in Ulysses 所使用的语法其实非常直观：

```
ulysses://x-callback-url/new-sheet?text=[[draft]]
```

可以看到，它遵循了 Ulysses URL Schemes 的标准结构。使用 `new-sheet` 动作，特别的是将 Drafts 中的文稿作为参数 `text` 的内容。这样将 Drafts 作为「草稿本」或者「速记本」，之后将文稿发送到 Ulysses，也不失为一种高效地写作流程。

利用 Sheet in Ulysses 整合初稿

类似的，在 Editorial 中也有 Markdown to Ulysses Workflow，可以将 Editorial 的 Markdown 文稿保存在 Ulysses 当中。

不过需要注意的是，Markdown to Ulysses 的作者在最后一步将 `group` 设为「Posts」，如果你没有对应的组，这个 Workflow 就无法正确运行。所以必须先修改为自己文库中存在的组，或者删掉 `group` 参数，选择将文稿保存进收件箱。

利用 Markdown to Ulysses 整合初稿

对于书写草稿的场景，我没有用 Launch Center Pro 来举例。熟悉这款应用的读者，在看过前文介绍 URL Schemes 时，想必已经知道该怎么在 LCP 中使用 Ulysses 的 URL Schemes 来添加草稿。

但是鉴于 LCP 的输入方式，我认为它并不适合和 Ulysses 协作，毕竟想在 LCP 大段输入文字，实在不是一种值得享受的体验。更多的时候，用 Launch Center Pro 来添加修改注释，是更自然的用法。

导出文稿时使用 URL Schemes

如果你发现 Ulysses 文库中已经积攒了大量应该归档的组和文章，可以将它们导出，让文库显得不那么臃肿。但是 Ulysses 中不支持一次导出整个文库，只能一篇篇手动导出。 Export Ulysses 整个 Workflow 就可以帮助你节约时间，一次性完成这个操作。

Export Ulysses 的作用，在于**批量导出 Ulysses 中的文稿到 iCloud Drive 或 Dropbox** 。这个 Workflow 由少数派作者真嗣编写，而且作者对导出后的文稿做了大量工作，使其既保持了原有结构，又能跳过文库中的空文稿。

Export Ulysses 实现的原理相对复杂，真嗣在《使用 Workflow 批量导出 Ulysses 文稿》一文中有详细的介绍，这里不再赘述。

利用 Export Ulysses 导出文库

小结

在 Mac 上，想要写作通常会肌肉记忆般的打开 Ulysses for Mac，再将其它工具的窗口作为辅助排在旁边。

而在 Ulysses for iOS，你可以先在 Safari 中随意的浏览资料，用 Web Page to Ulysses 摘录你觉得有价值的网页；在公交车上反复构思文章的结构编排，通过 Launch Center Pro 来记下灵感；在 Drafts 中初拟草稿，随后再转到 Ulysses 中仔细打磨。

简而言之，URL Schemes 给了 Ulysses for iOS 更多自由。

Ulysses for iOS 虽然大体上秉承了 Ulysses for Mac 的设计和功能，不过利用 iOS 的诸多特性，带来了不少 iOS 上才有的体验。而对 URL Schemes 的支持，则赋予了它更多可能性。

至此，我们对 Ulysses 本身在 iOS 和 macOS 上的介绍基本结束。经过这四章的介绍，Ulysses 作为「写作工具」的种种优势，已经展示的淋漓尽致。

而接下来的教程，我们将把眼光放的更广一些，期待 Ulysses 不仅能写，而且能真正在各种场合中，导出合适样式的文稿。

我们下一章见！